

Managing Distributed MARF with SNMP

Serguei A. Mokhov
Lee Wei “Lewis” Huynh
Jian “James” Li

Concordia University
Montréal, Québec, Canada

Tue 2 Jun 2009 05:22:48 EDT

Contents

1	Introduction	2
1.1	Background	2
1.2	Scope	5
1.3	Tools	6
1.4	Summary	6
2	Methodology	9
2.1	Introduction	9
2.2	MARF-Manager-Agent Architecture	10
2.3	SMI Structure	10
2.4	MARF Services	14
2.4.1	General Service MIB	15
2.4.2	Storage	15
2.4.3	Sample Loading	15
2.4.4	Preprocessing	15
2.4.5	Feature Extraction	20
2.4.6	Classification	20
2.4.7	Applications	24

3	Conclusion	26
3.1	Review of Results	26
3.1.1	MIBs	26
3.1.2	SNMP Proxy Agents	27
3.1.3	SNMP MARF Application Managers	27
3.1.4	Difficulties	28
3.1.5	Contributions	30
3.1.6	Open Source	30
3.2	Future Work	31
3.2.1	Scenarios	31
3.2.2	Summary	32
3.3	Acknowledgments	33
	Bibliography	34

List of Figures

1.1	The Core MARF Pipeline Data Flow	3
1.2	The Distributed MARF Pipeline	4
1.3	SpeakerIdenApp Client GUI Prototype (Manager)	7
1.4	MARF Service Status Monitor GUI Prototype (Agent)	8
2.1	MARF-Manager-Agent Architecture	11
2.2	Preliminary MARF Private Enterprises Number.	12
2.3	Preliminary MARF General Tree.	13
2.4	General Service MIB 1.	16
2.5	General Service MIB 2.	17
2.6	Storage MIB.	18
2.7	Preliminary MARF Sample Loading Service MIB.	19
2.8	Preliminary MARF Preprocessing Service MIB.	21
2.9	Preliminary MARF Feature Extraction Service MIB.	22
2.10	Preliminary MARF Classification Service MIB.	23
2.11	SpeakerIdenApp MIB.	24
2.12	LangIdenApp MIB.	25

Chapter 1

Introduction

Revision : 1.1.2.6

1.1 Background

The Modular Audio Recognition Framework (MARF) [The09, MCSN03, Mok06] is an open-source research platform and a collection of voice, sound, speech, text, and natural language processing (NLP) algorithms written in Java and arranged into a modular and extensible framework facilitating addition of new algorithms. MARF can run distributedly over the network (using CORBA, XML-RPC, or Java RMI) and may act as a library in applications or be used as a source for learning and extension. A few example applications are provided to show how to use the framework. One of MARF's applications, **SpeakerIdentApp** has a database of speakers, where it can identify who people are regardless what they say.

Original MARF [MCSN03] was developed by Serguei Mokhov with a few classmates and others throughout a variety of courses. Distributed MARF [Mok06] (DMARF) proof-of-concept (PoC) implementation was done by Serguei Mokhov in the Distributed Systems class. The Distributed MARF nodes are hard to manage if there are many, including configuration, statistics, and status management.

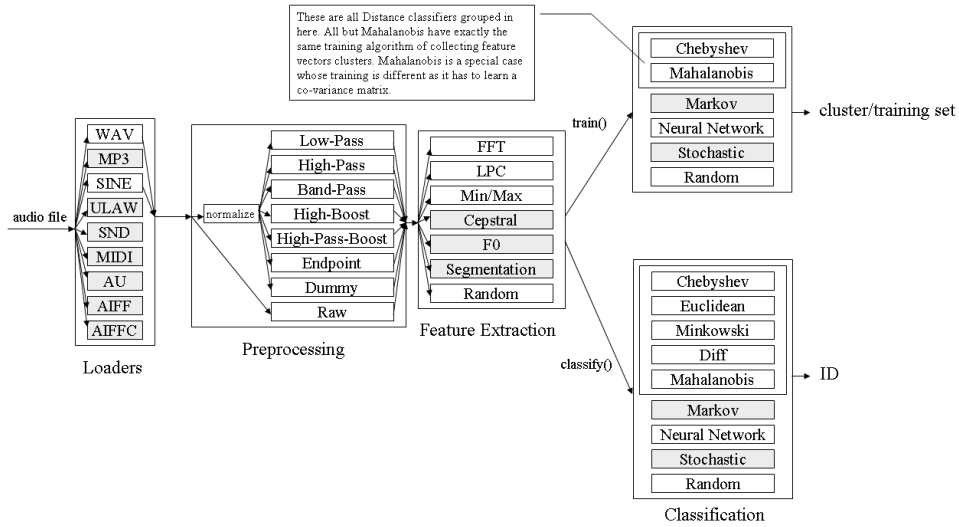


Figure 1.1: The Core MARF Pipeline Data Flow

MARF has several applications. Most revolve around its recognition pipeline – sample loading, preprocessing, feature extraction, and training or classification. One of the applications, for example is Text-Independent Speaker Identification. In the classical MARF, the pipeline and the applications as they stand are purely sequential with even little or no concurrency when processing a bulk of voice samples. Thus, the purpose of DMARF in [Mok06] was to make the pipeline distributed and run on a cluster or a just a set of distinct computers to compare with the traditional version and add disaster recovery and service replication, communication technology independence, and so on.

The classical MARF’s pipeline is in Figure 1.1. The goal of DMARF was to distribute the shown stages of the pipeline as services as well as stages that are not directly present in the figure – sample loading, front-end application service (e.g. speaker identification service, etc.) among other things in the distributed system. The reasons to be able flexibly distribute these services is to offload the bulk of multimedia/data crunching and processing to a higher performance servers, that can communicate, while the data collection may happened at several low-cost computers (e.g. low-end laptops) or PDAs

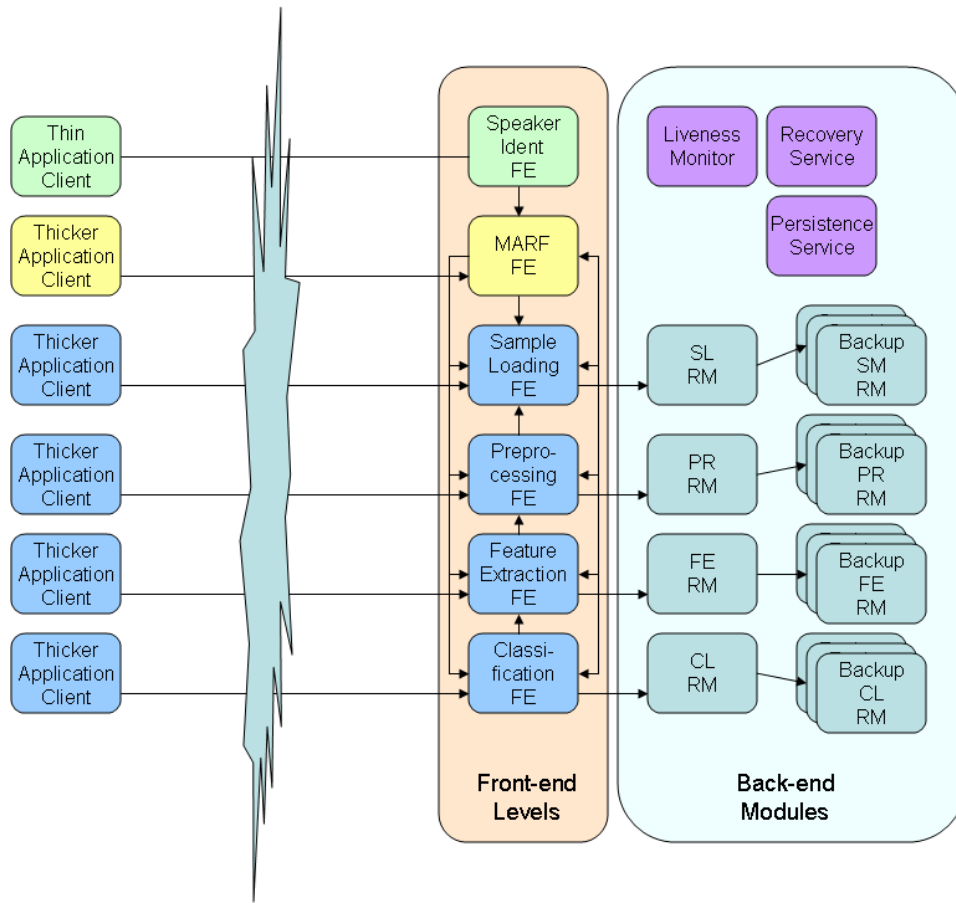


Figure 1.2: The Distributed MARF Pipeline

alike, embedded devices, etc., which may not necessarily have the processing power and storage capacity locally to cope with the amount of incoming data, so they pass it on to the servers. (A possible infrastructure of such a setup can, for example, be in place in different law enforcement agencies spread out across a country, yet, being able to identify speakers across all jurisdictions if say recorded phone conversations of a suspect are available. In another scenario, the could be used in tele-conferencing.)

In Figure 1.2 the distributed version of the pipeline is presented. It indicates different levels of basic front-ends, from higher to lower, which a client application may invoke as well as services may invoke other services through their front-ends while executing in a pipeline-mode. The back-ends are in charge of providing the actual servant implementations as well as the features like primary-backup replication, monitoring, and disaster recovery modules.

The status management graphical user interface (GUI) prototypes were developed in DMARF, but not implemented to provide application managers an ability to monitor services' status (or the whole pipeline) as well as change configuration options, as in Figure 1.3 and in Figure 1.4.

1.2 Scope

The scope of this project's work focuses on the research and prototyping of the extension of the Distributed MARF such that its services can be managed through the most popular management protocol familiarly, SNMP. The rationale behind SNMP vs. MARF's proprietary management protocols, is that can be integrated with the use of common network service and device management, so the administrators can manage MARF nodes via a already familiar protocol, as well as monitor their performance, gather statistics, set desired configuration, etc. perhaps using the same management tools they've been using for other network devices and application servers.

MARF has generally thee following type of services: application, core pipeline, sample loading, preprocessing, feature extraction, and classification. There are common data structures, configuration, storage management attributed to them. DMARF's components in general are stand-alone and may listen on RMI, XML-RPC, CORBA, and TCP connections for their needs, and natively do not "understand" SNMP. Therefore, each managed service will have to have a proxy SNMP-aware agent for management tasks and delegate instrumentation proxy to communicate with the service's specifics. Thus, in this work we are designing and implementing to some extent the following:

- Defining MIBs for MARF Services
- Producing Proxy SNMP Agents
- Agent-Adapter Delegate Instrumentation
- SNMP MARF Manager Applications

The original proposal has a lot more provisional tasks, that are outlined in Section 3.2.

1.3 Tools

We will use platform-independent tools like Java, possibly JDMK based on JMX, eventually RMI, CORBA, XML-RPC as provided by AdventNet or others. SimpleWeb [The07] for original MIB cross-validation and AdventNet's SNMP Java API [Adv07b] and Java Agent SDK [Adv07a] tools are in actual active use.

1.4 Summary

The project seems like a viable and useful option to extend MARF and contribute its implementation at the same time to the open-source community.

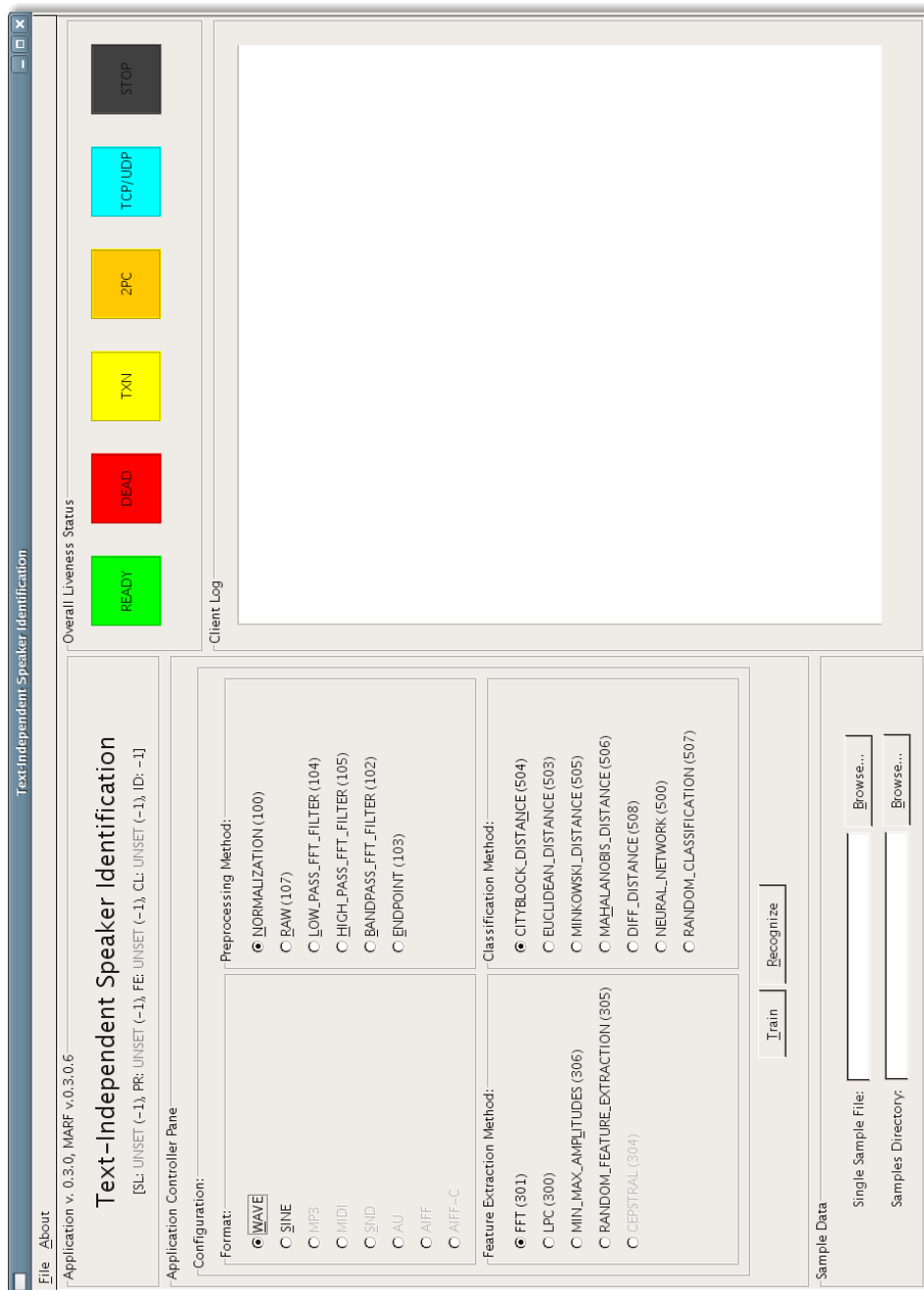


Figure 1.3: SpeakerIdenApp Client GUI Prototype (Manager)

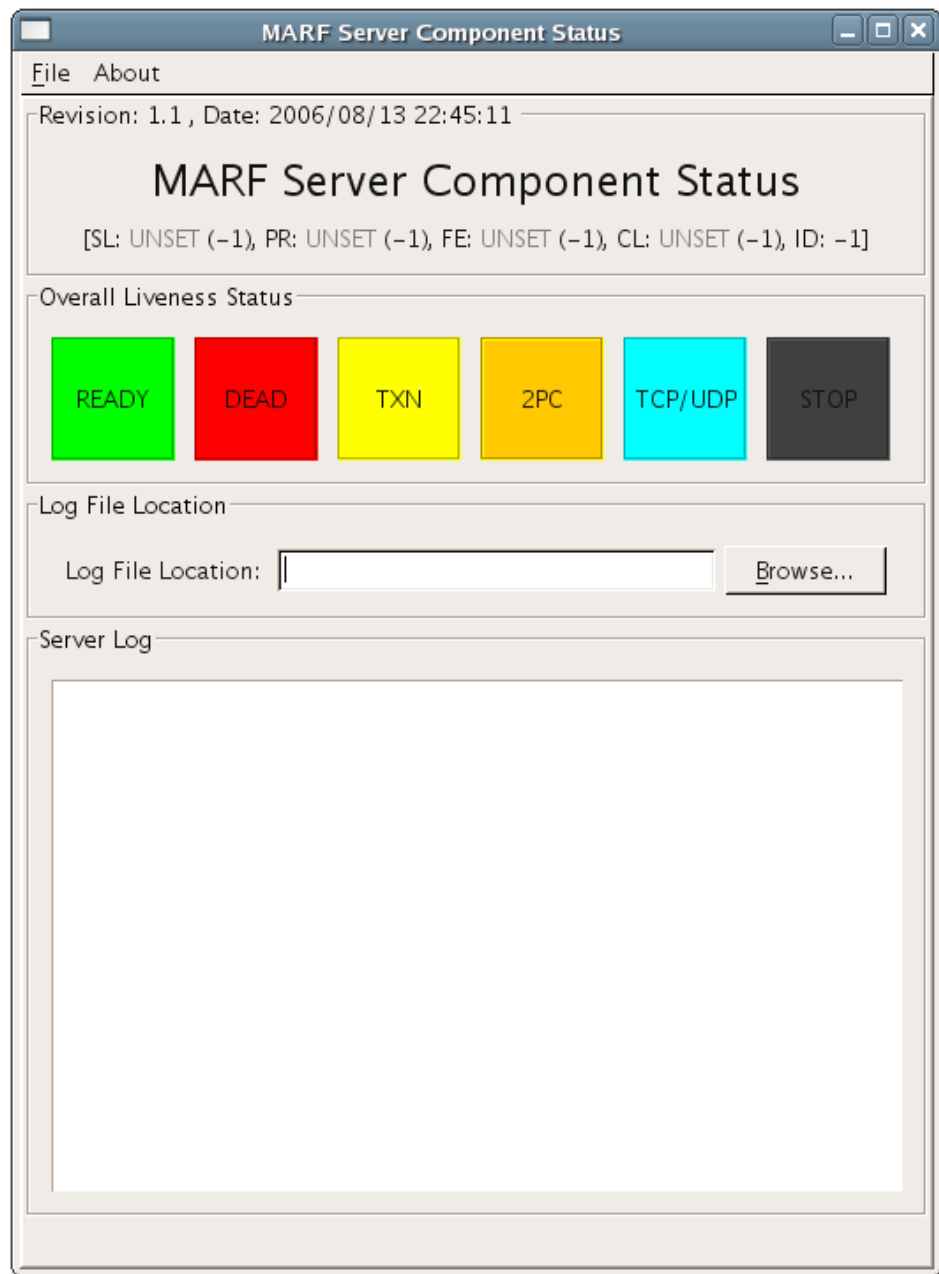


Figure 1.4: MARF Service Status Monitor GUI Prototype (Agent)

Chapter 2

Methodology

Revision : 1.1.2.10

2.1 Introduction

Distributed MARF [The09] offers a number of service types:

1. Application Services
2. General MARF Pipeline Services
3. Sample Loading Services
4. Preprocessing Services
5. Feature Extraction Services
6. Training and Classification Services

which are backed by the corresponding server implementations in CORBA, Java RMI, and Web Services XML-RPC. The services can potentially be embedded into other application or hardware systems for speaker and language identification, and others.

We are interested in managing such services over a network as a whole, collect their processing statistics, perform remote management, etc., so we need to define them in MIB using ASN.1 including the types of requests and responses and their stats.

We have applied to IANA [(IA07)] and registered a private enterprise number (PEN) under the `enterprises` node, which is 28218 (under the MARF Research and Development Group).

2.2 MARF-Manager-Agent Architecture

We devised a preliminary management architecture for MARF application and services, presented in Figure 2.1. In this figure we capture a relationship between all the major entities in the system. Application are the ultimate managers, whereas, the remaining services can be both managers and agents in some cases. The MARF service which operates the pipeline on behalf of main applications, can manage sample loading, preprocessing, feature extraction, and classification. Since those services can talk to each other and request data from each other (e.g. classification can request data from feature extraction), they may exhibit manager characteristics, not fully explored in this work. Applications don't need to come through the MARF service manager, but in case some need arises (debugging, development, maintenance), can connect to the terminal services of the pipeline directly.

2.3 SMI Structure

We have worked on the MIB tree for the entire DMARF and did some progress in defining general services and storage types, Preprocessing, Feature Extraction, Classification, SpeakerIdentApp, and LangIdentApp MIBs, which are under the `marf/src/mib` directory. We started off with `WWW-MIB` [HKS99] as an example for a `WwwService`'s and some works from `ATM-TC-MIB` and related files from the same source. We preliminary completed the indicated MIBs and provided default proxy implementations for the services,

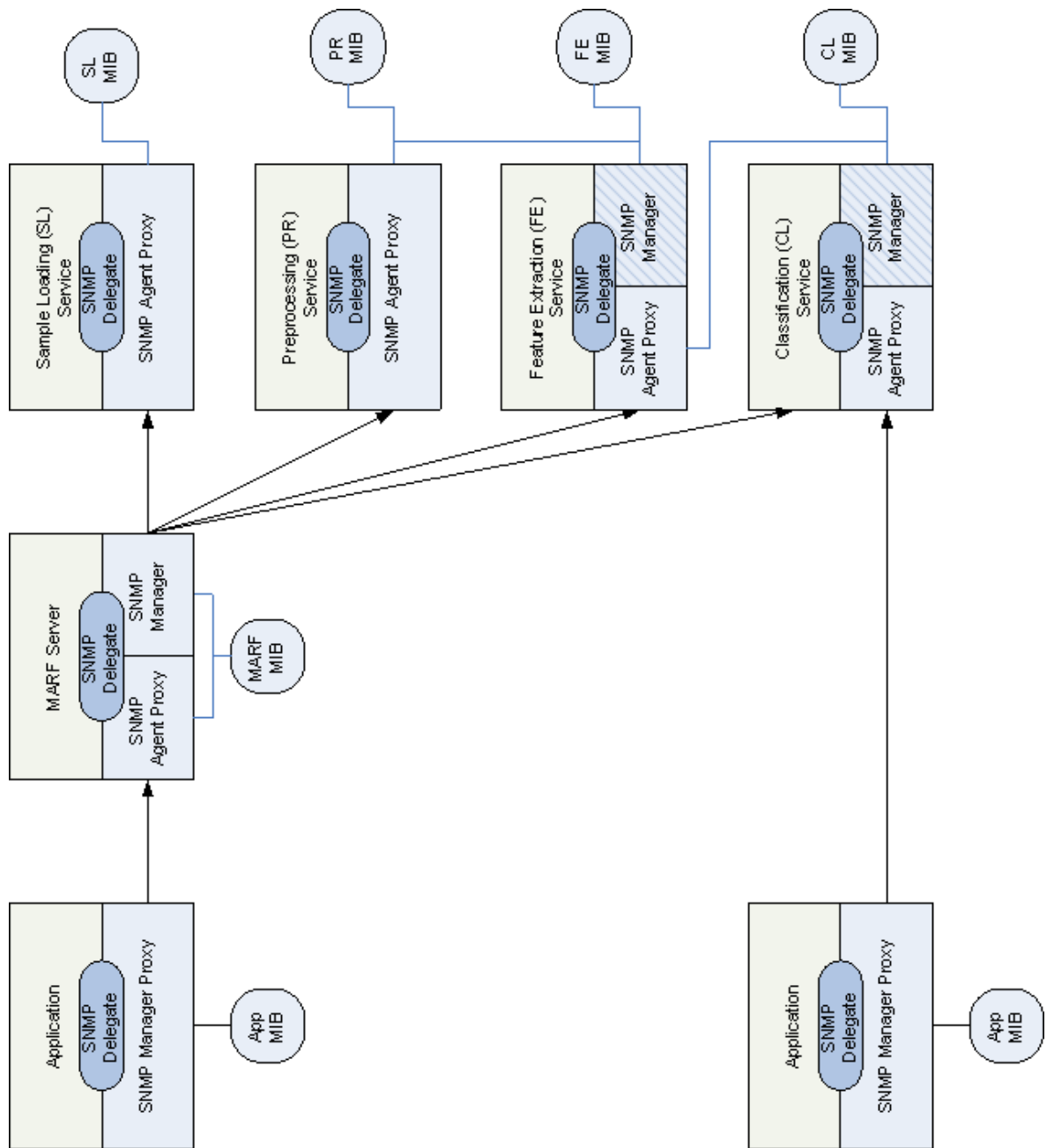


Figure 2.1: MARF-Manager-Agent Architecture

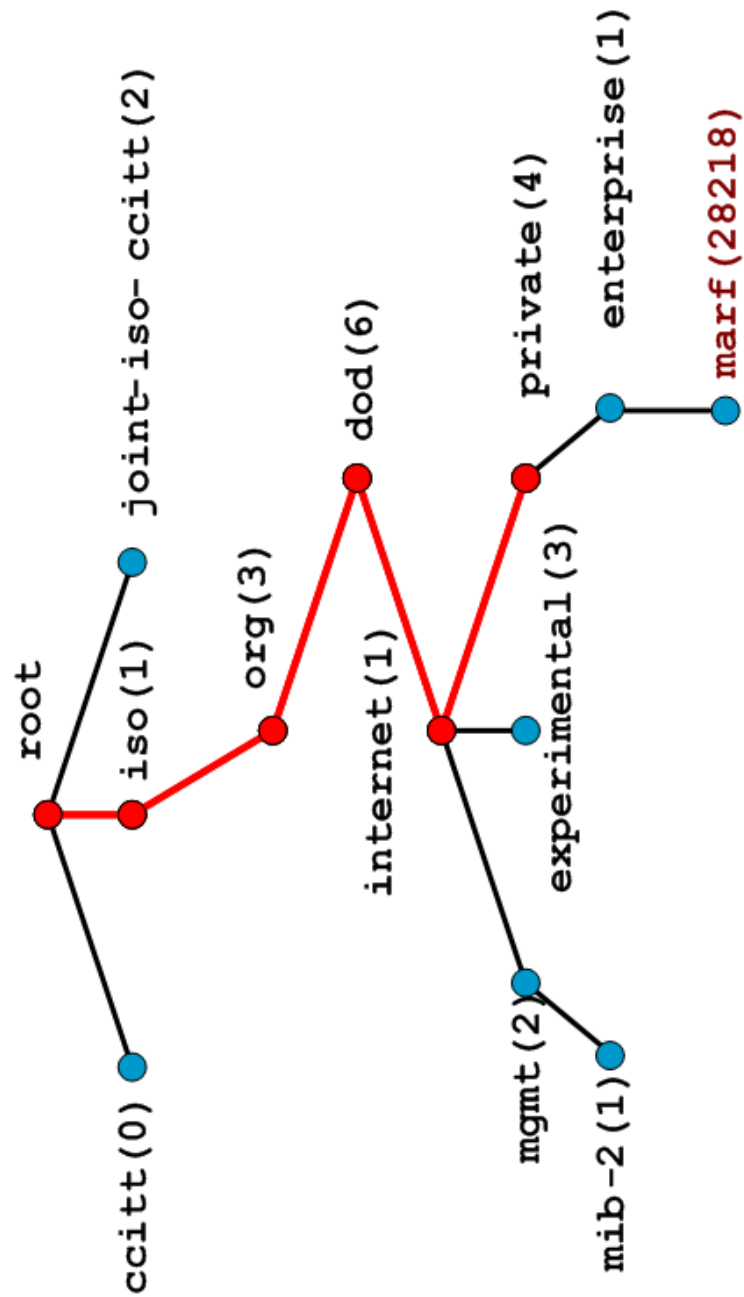


Figure 2.2: Preliminary MARF Private Enterprises Number.

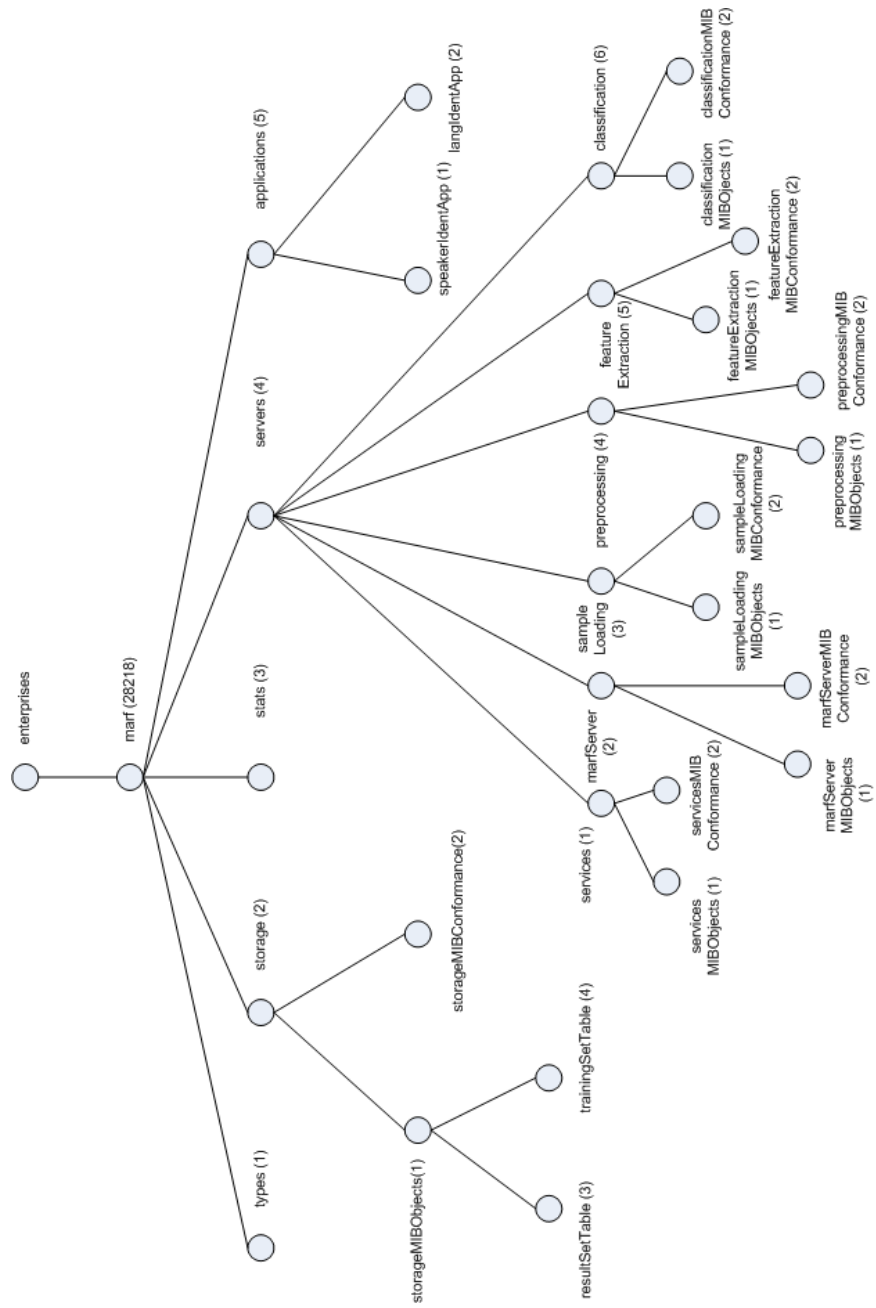


Figure 2.3: Preliminary MARF General Tree.

at least a few of them using the AdventNet’s API and SDK, which can be found under `marf/src/marf/net/snmp` directory tree. In Figure 2.2 we show where MARF’s MIB subtree begins under the `enterprises` node. And in Figure 2.3 we present the general overview of the most SMI components for MARF services and other required components. We also used lecture notes of Dr. Chadi Assi [Ass07]. We primarily using SNMPv2 along with SMIV2 in this project.

2.4 MARF Services

This section provides a sneakpeek on the MARF service types we are dealing with and some of them have their preliminary MIB subtrees drawn. As mentioned earlier we provide our MIBs so far in `marf/src/mib`. Note, some of the diagrams may not correspond 1-to-1 to MIBs as it is work in progress and things sometimes get out of sync. The relevant files are (as of this writing):

- `MARF-MIB.mib` – main file meant to consolidate all when ready
- `MARF-types.mib` – some common textual conventions
- `MARF-storage.mib` – storage-related issues and types so far
- `MARF-services.mib` – general services description
- `MARF-sample-loading.mib` – concrete sample loading service
- `MARF-preprocessing.mib` – concrete preprocessing service
- `MARF-feature-extraction.mib` – concrete feature extraction service
- `MARF-classification.mib` – concrete classification service
- `MARF-APPS-SPEAKERIDENTAPP.mib` – a MIB for SpeakerIdentApp
- `MARF-APPS-LANGIDENTAPP.mib` – a MIB for LangIdentApp

2.4.1 General Service MIB

Most MARF services share some common definitions about indexing the services, their statistics, and so on, so the general service module was created such that most of this general functionality is captured in there including statistics, and the more specific modules extend by augmenting its tables and using its types. The generic service description is in Figure 2.5.

2.4.2 Storage

MIB for storage related activities (e.g. training sets, classification results, etc.) has to be provided, as such the MIB presented in Figure 2.6 was devised.

2.4.3 Sample Loading

The Sample Loading Service knows how to load certain file or stream types (e.g. WAVE) and convert them accordingly for further preprocessing. In our project, we are introducing the sample loading module, which we will use to manage and keep tracking its specific parameters which are:

1. **iFormat**: a sample format, an integer type attribute.
2. **adSample**: sample data, a collection of bytes.

All these attributes will be located in **sampleLoadingServiceEntry** object which is in **sampleLoadingServiceTable** object. See the example of the SMI tree for sample loading in Figure 2.7.

2.4.4 Preprocessing

The Preprocessing service accepts incoming voice or text samples and does the requested preprocessing (all sorts of filters, normalization, etc.). Its function generally is to normalize the incoming sample file by amplitude starting from certain index and/or filter it. It has several algorithms to be

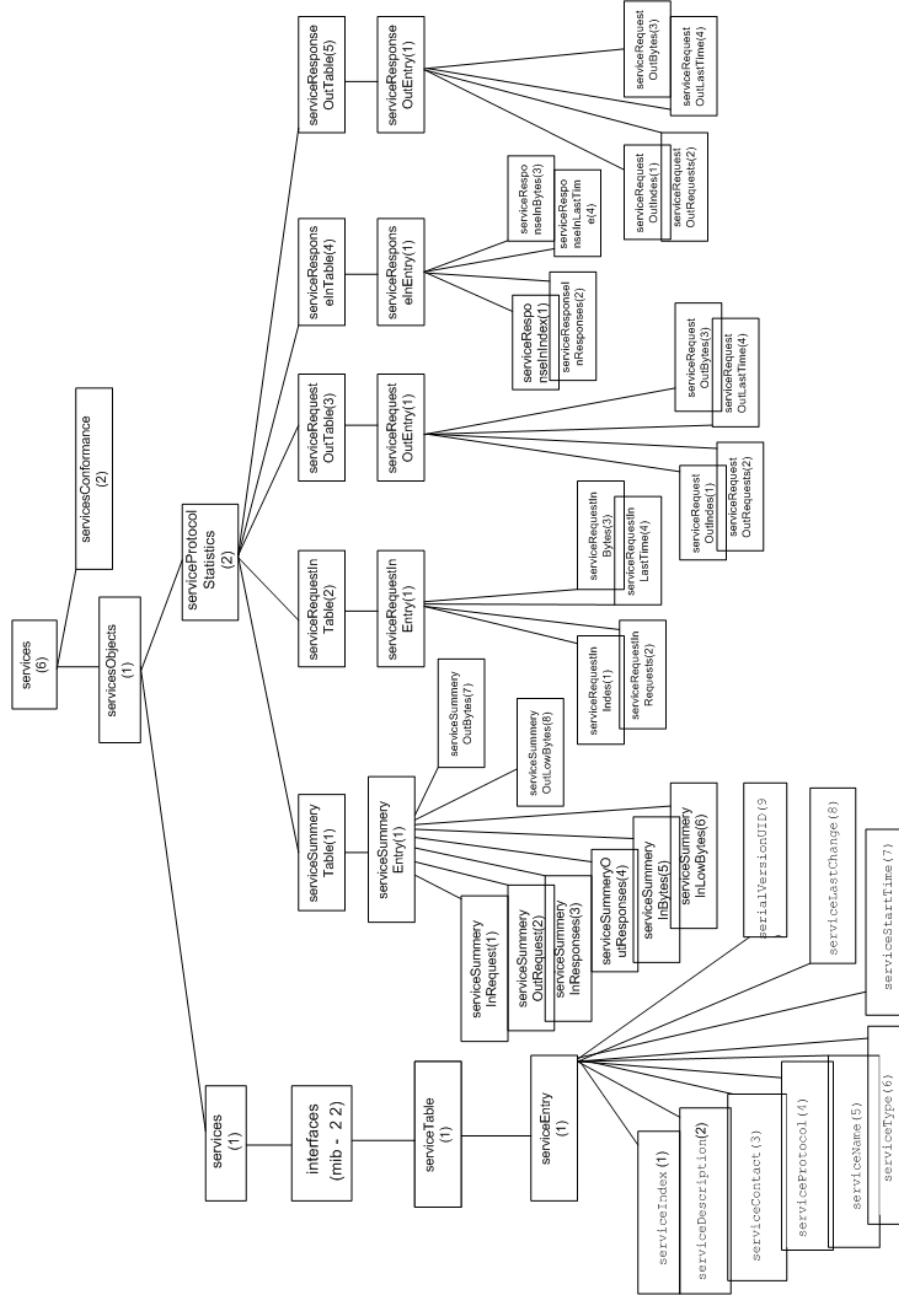


Figure 2.4: General Service MIB 1.

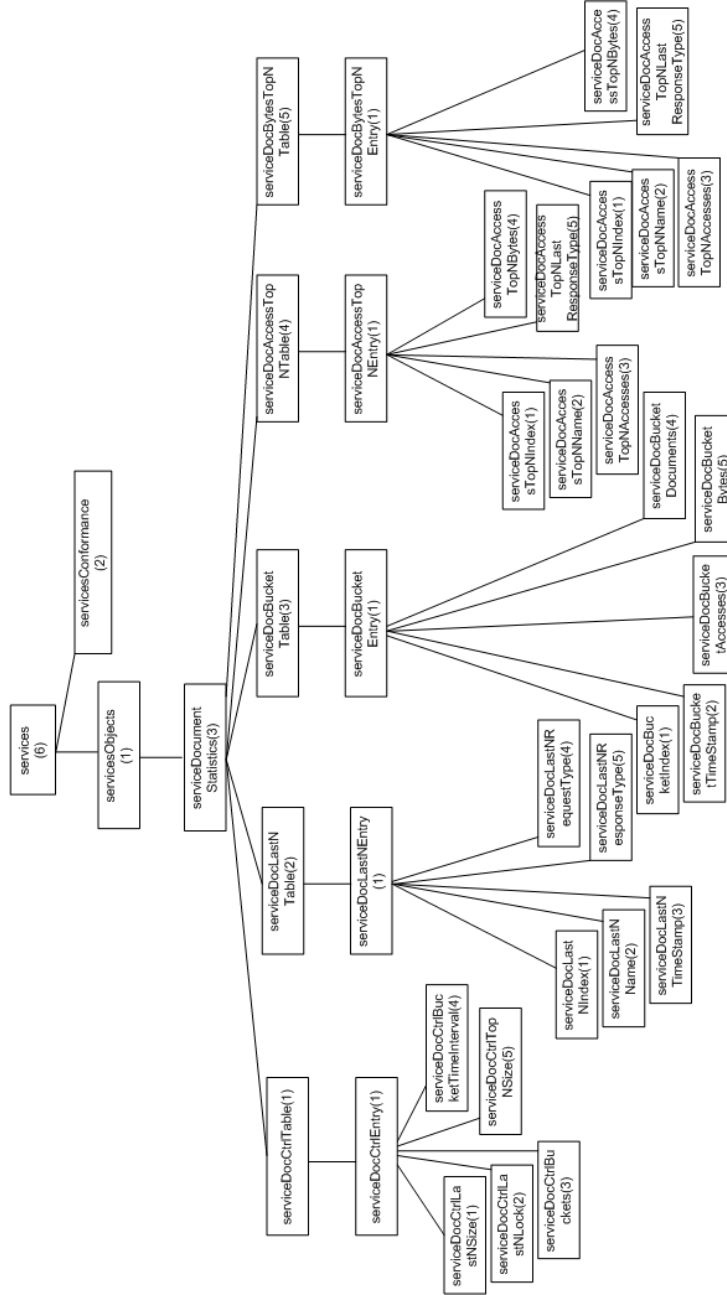


Figure 2.5: General Service MIB 2.

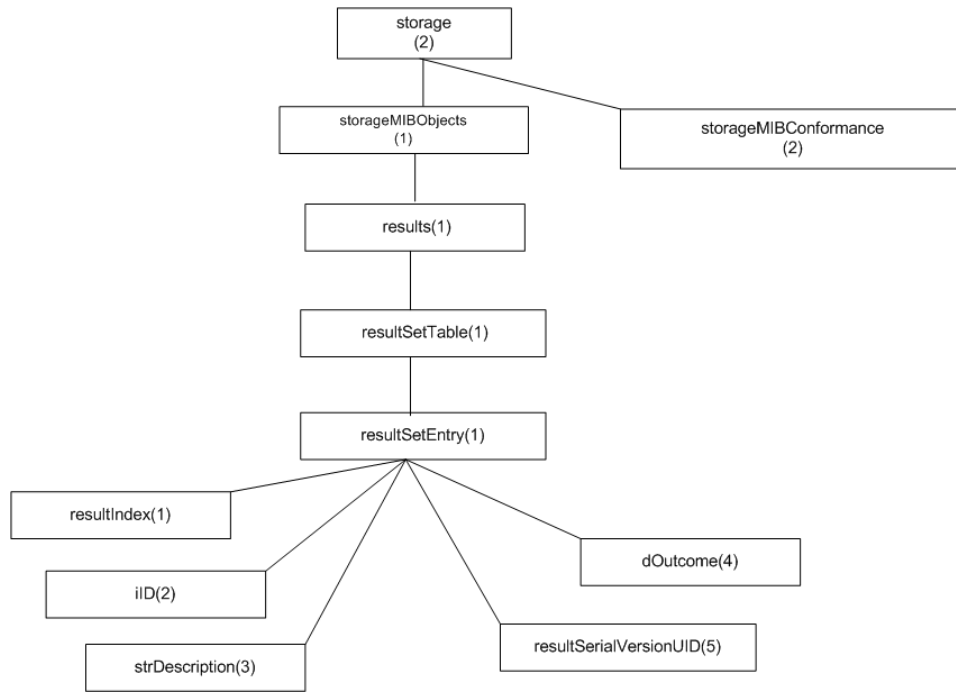
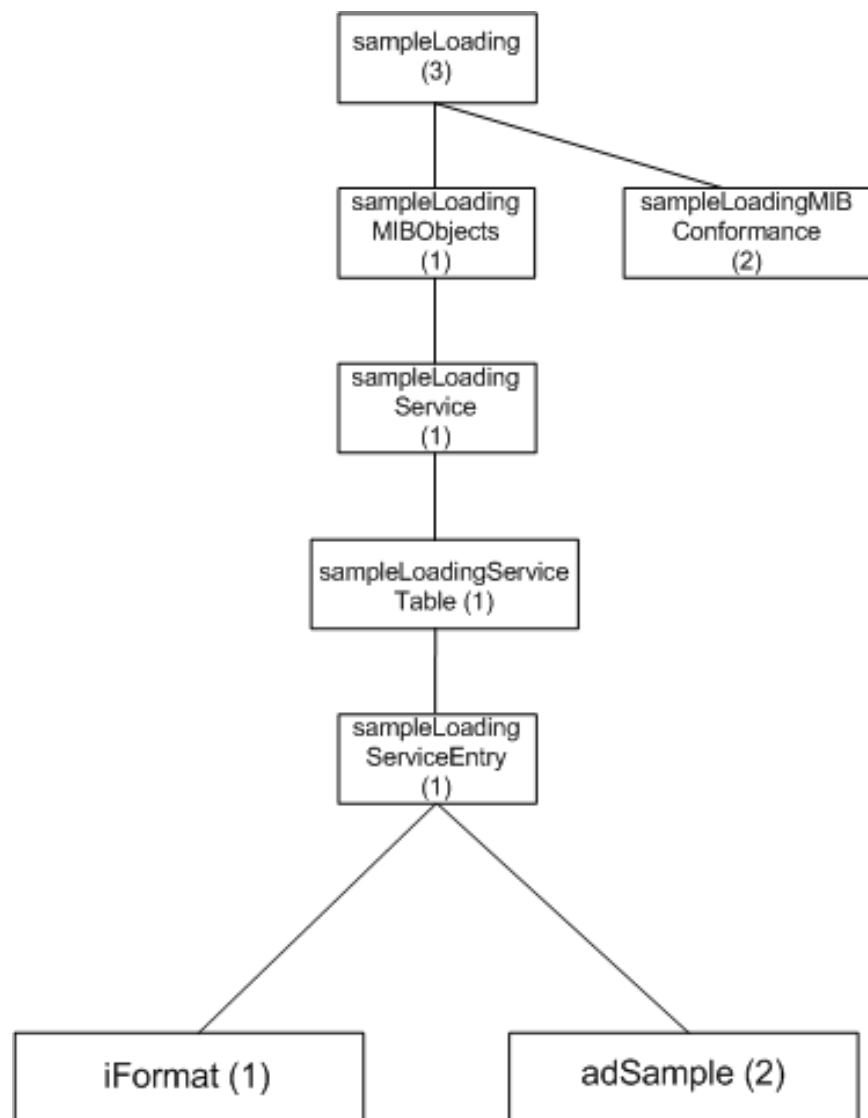


Figure 2.6: Storage MIB.



sampleLoading Module

Figure 2.7: Preliminary MARF Sample Loading Service MIB.

as an option to filter the voice frequencies. The algorithms in this module are FFT-based and CFE-based. In our project, we introduce the preprocessing MIB module, which will use to manage and keep tracking its specific parameters which are:

1. **Sample**: a collection of doubles and a format.
2. **dSilenceThreshold** : a double type for the silence cut off threshold.
3. **bRemoveNoise** : a Boolean type to indicate noise removal.
4. **bRemoveSilence** : a Boolean type to indicate silence removal.

All these attributes are located in the **preprocessingServiceEntry** object which is in **preprocessingServiceTable** tabular object. See the example of the SMI tree for preprocessing in Figure 2.8.

2.4.5 Feature Extraction

Feature Extraction service accepts data, presumably preprocessed, and attempts to extract features out of it given requested algorithm (out of currently implemented, like FFT, LPC, MinMax, etc.) and may optionally query the preprocessed data from the Preprocessing Service. See the SMI tree for feature extraction in Figure 2.9. The parameters been used in the Feature Extraction are below:

1. **adFeatures**: a currently being processed feature vector.
2. **oFeatureSet**: a collection of feature vectors or sets for a subject.

2.4.6 Classification

Classification-type of services are responsible for either training on features data (read/write) or classification (read-only) of the incoming data. Being a bit similar in operation to the service types mentioned earlier, Classification



Preprocessing Module

Figure 2.8: Preliminary MARF Preprocessing Service MIB.

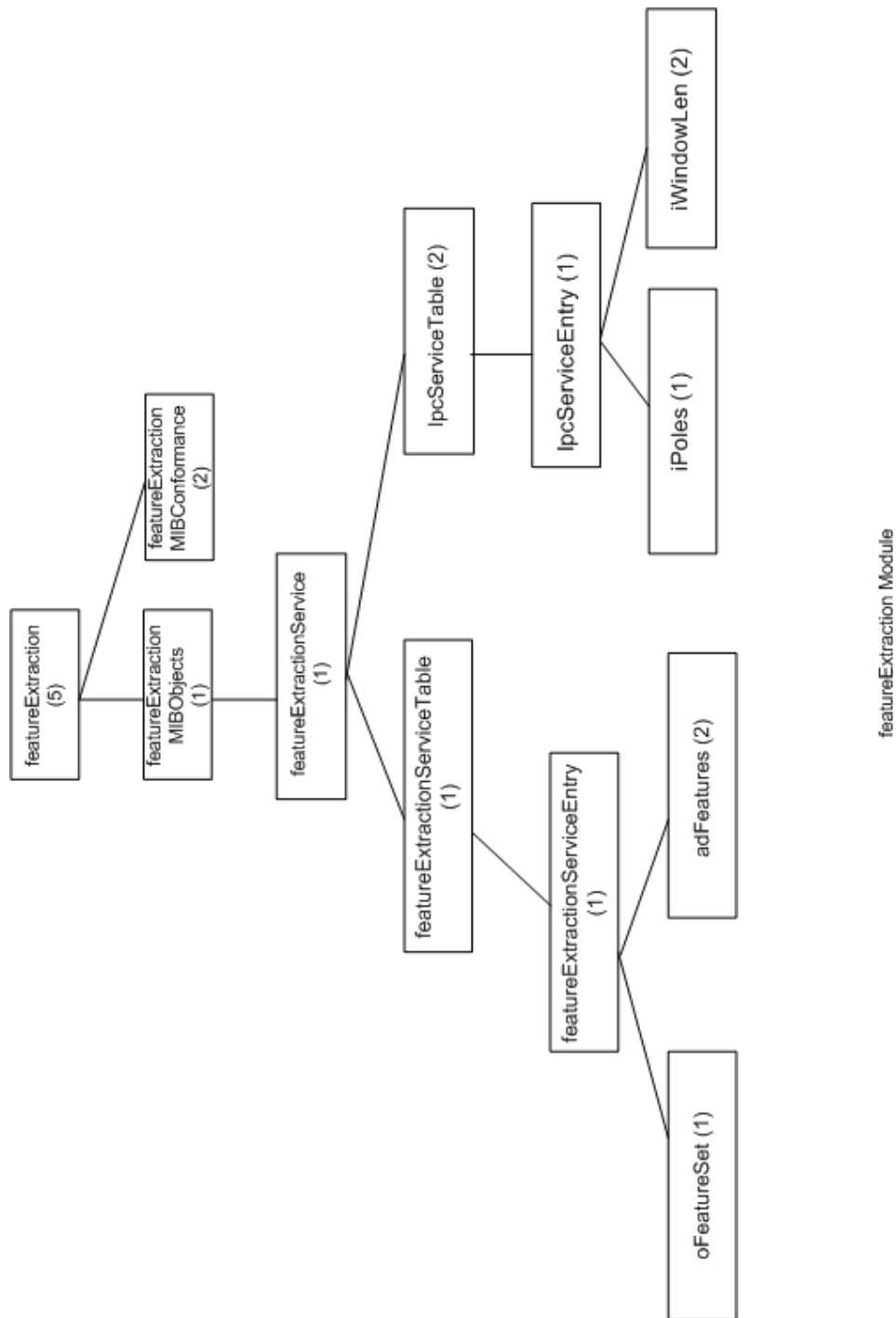


Figure 2.9: Preliminary MARF Feature Extraction Service MIB.

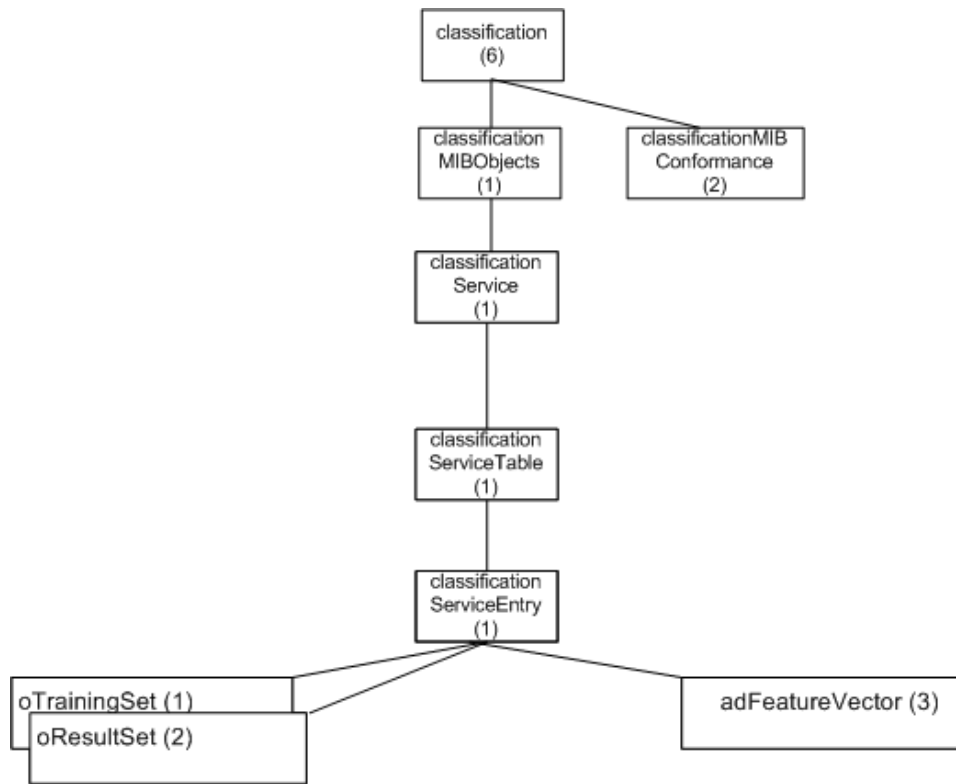


Figure 2.10: Preliminary MARF Classification Service MIB.

does a lot more legwork and is actually responsible saving the training set database (in files or otherwise) locally, provide concurrency control, persistence, ACID, etc. Its MIB tree in Figure 2.10 is very similar to the other services at this point.

1. **adFeatures:** a currently being processed feature vector for training/classification.
2. **oResultSet:** a collection of classification results.

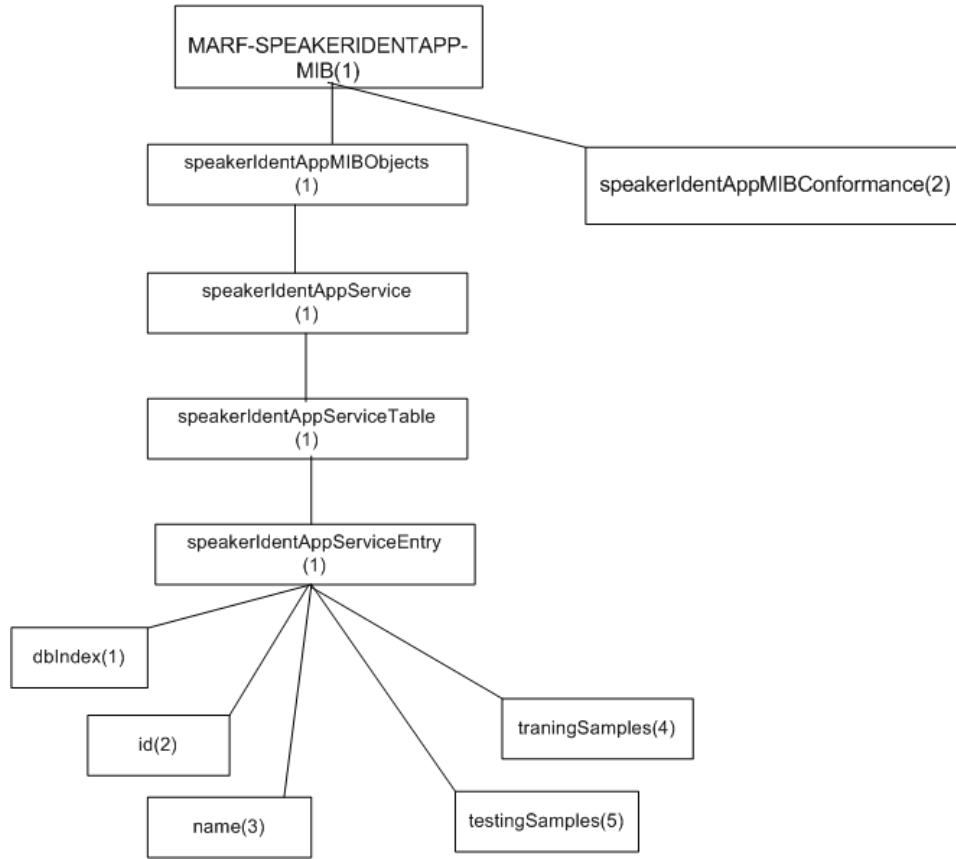


Figure 2.11: SpeakerIdentApp MIB.

2.4.7 Applications

MARF has many applications, for this project we were considering primarily two applications for management, SpeakerIdentApp as well as LangIdentApp for speaker and language identification. The MIB trees we designed are in in Figure 2.11 and Figure 2.12.

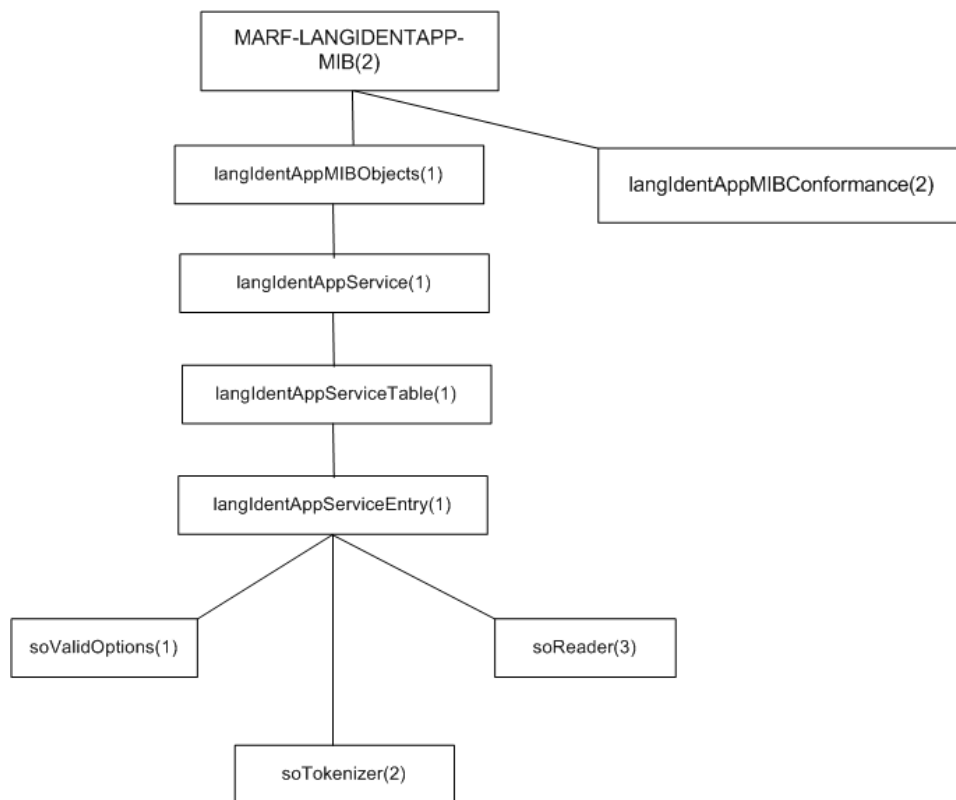


Figure 2.12: LangIdentApp MIB.

Chapter 3

Conclusion

Revision : 1.1.2.5

3.1 Review of Results

3.1.1 MIBs

By far, we have successfully finished a set of compilable and loadable MIBs from different aspects of DMARF over SNMP:

1. MARF-MIB
2. MARF-types
3. MARF-storage
4. MARF-services
5. MARF-sample-loading
6. MARF-preprocessing
7. MARF-feature-extraction
8. MARF-classification

9. MARF-APPS-SPEAKERIDENTAPP

10. MARF-APPS-LANGIDENTAPP

While some of the information in the above MIBs is in debug form, it is enough to compile and generate proxy agent for minimal testing. There are some unused definitions that will either be removed or become used in the follow up revisions. We also obtained a PEN SMI number of 28218 from IANA for use in MARF.

3.1.2 SNMP Proxy Agents

Based on the MIBs above, we produced proxy agents, using AdventNet's tools [Adv07a]. There are two kinds of proxies: one is proxy talking to the MARF's API and the agent, we call it *API proxy* or *instrumentation proxy*. The other one is proxy talking to manager and agents by using SNMP. Because the time constraints, they are not fully instrumented.

Agents produced with the help of MIB Compiler of AdventNet also produces two types of SNMP agents master agent (proxy to a group of SNMP sub-agents that manager does not see) vs. sub-agents (SNMP) as well as instrumentation (delegates, application-specific business logic). E.g. in the case of Feature Extraction and LPC, the former would be the master agent proxy, and LPC would be a sub-agent (e.g. a MIB subtree), which is a more specific type of feature extraction.

3.1.3 SNMP MARF Application Managers

Within the timeframe of the course we did not manage to produce our own SNMP manager application part and integrate it into the proposed GUI, so we are leaving this task to the future work. The manager application we used to test our work was the MIB Browser provided by the AdventNet's tools.

3.1.4 Difficulties

During our design, MIB validation, and compilation, we faced certain difficulties. One of them the difference in ASN.1/SMI syntax checks between tools such as SimpleWeb vs AdventNet, where we could not debug for long time one problem we faced: double AUGMENTS when we do the MIB of general an concrete MARF services (e.g. Feature Extraction to LPC or Classification to Neural Network, etc.). MIB loading and configuration management in AdventNet and the corresponding mapping of operations (instrumentation delegates in a pipeline) were a part of the learning curve for AdventNet’s API.

Here is the example that was holding us back where SimpleWeb’s validator [The07] complained, but the AdventNet’s MIB Browser and Compiler didn’t have problems with. Assume:

```
tableFoo
tableEntryFoo

tableBar
tableEntryBar
    AUGMENTS {tableEntryFoo}
```

SimpleWeb’s validator complained here:

```
tableBaz
tableEntryBaz
    AUGMENTS {tableEntryBar}
```

To give a more concrete example from one of our MIBs (a bit stripped). In this case `lpcServiceEntry` would be managed by a sub-agent of feature extraction. Please see more up-to-date MIB in `MARF-feature-extraction.mib`.

```
featureextractionServiceTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF FeatureextractionServiceEntry
```

```

MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The table of the Featureextraction services known by the SNMP agent."
AUGMENTS { serviceTable }
::= { featureextractionService 1 }

featureextractionServiceEntry OBJECT-TYPE
    SYNTAX FeatureextractionServiceEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "Details about a particular Featureextraction service."
    AUGMENTS { serviceEntry }
    ::= { featureextractionServiceTable 1 }

FeatureextractionServiceEntry ::= SEQUENCE {
    oFeatureSet FeatureSet,
    adFeatures VectorOfDoubles
}

lpcServiceTable OBJECT-TYPE
    SYNTAX SEQUENCE OF LPCServiceEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION " "
    AUGMENTS { featureextractionServiceTable }
    ::= { featureextractionService 2 }

lpcServiceEntry OBJECT-TYPE
    SYNTAX LPCServiceEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION " "
    AUGMENTS { featureextractionServiceEntry }
    ::= { lpcServiceTable 1 }

```



```

LPCServiceEntry ::=SEQUENCE {
    iPoles    INTEGER,
    iWindowLen INTEGER
}

```

3.1.5 Contributions

Generally, the entire team has focused on the development of the MIBs, learning and trying out AdventNet's API, project presentation, and report equally. Since MARF consists of multiple components, the workload was subdivided roughly along those components, some common structures were worked on together by the team. Some specific breakdown is as follows:

- Serguei: overall project design and management, PEN application, MIB compilation, Classification and MARF server MIBs.
- Jian: learning MARF's API, Feature Extraction, LPC MIBs, AdventNet's MIB Compiler and API investigation, MIB diagrams.
- Lee Wei: learning MARF's API, Sample Loading, Preprocessing, Application MIBs, AdventNet's MIB Compiler and API investigation.

3.1.6 Open Source

This project, as the MARF itself, was developed as open-source project as **SourceForge.net**. To checkout latest version the source code, this report sources, MIBs, and the presentation from our CVS repository, one can do by executing the following commands:

```

cvs -d:pserver:anonymous@marf.cvs.sourceforge.net:/cvsroot/marf login
cvs -z3 -d:pserver:anonymous@marf.cvs.sourceforge.net:/cvsroot/marf co -rINSE7120 -P marf

```

or alternatively browse it on-line:

<http://marf.cvs.sourceforge.net/marf/marf/?pathrev=INSE7120>

The files of the interest related to this project can be found as follows: this report sources are in `marf/doc/src/tex/inse7120`, the presentation is in `marf/doc/presentations/inse7120`, the corresponding graphics is in `marf/doc/src/graphics/distributed/mib`, the source code of the entire MARF is in `marf/src`, and the generated agent code is in `marf/src/marf/net/snmp`.

3.2 Future Work

This section summarizes future work highlights.

3.2.1 Scenarios

Here we present a few scenarios where DMARF along with the SNMP management could be usefully employed. Management of these infrastructures is better be conducted over SNMP, which is well adapted for the use of networks under stress, along with RMON, and configuration management by some kind of central authority.

- Police agents in various law enforcement agencies spread out across a country, yet, being able to identify speakers across all jurisdictions if say recorded phone conversations of a suspect are available.
- Assume another scenario for conference microphones for different speakers installed in a large room or separate rooms or even continents using teleconferencing. We try to validate/identify who the speakers are at a given point in time.
- Alternatively, say these are recorded voices of conference calls or phone conversations, etc. We could have it reliable, distributed, with recovery of agents/clients and the server database.
- Perhaps, other multimedia, VoIP, Skype, translation and interpretation natural language services can be used with DMARF over the Internet and other media.

3.2.2 Summary

There is a lot of distributed multimedia traffic and computation involved between sample loading, preprocessing, feature extraction, and classification servers. Efficiency in network management involves avoidance of computations that already took place on another server and just offloading the computed data over.

Since DMARF's originally implemented the Java RMI, SOAP (Web-Services over XML-RPC), and CORBA, some of that work can be transplanted towards the management needs. In particular, DMARF's CORBA IDL definitions can as well be used for SNMP agent generation. Thus, in this project another area to focus will be on the role of CORBA in network management and the extension of Distributed MARF's [Mok06] CORBA services implementation and its SpeakerIdentApp to provide efficient network management, monitoring, multimedia (audio) transfer, fault-tolerance and recovery, availability through replication, security, and configuration management.

More specifically, in the context of MARF we'll look into few more of the aspects, based on the need, time, and interest. Some research and implementation details to amend Distributed MARF we will consider of the following:

- Finish proxy agents and instrumentation.
- Implement our own managers and the functions to compile new MIBs into the manager.
- Complete prototyped GUI for ease-of-use of our management applications (as-is MARF is mostly console-based).
- Complete full statistics MIB and implement RMON along with some performance management functions such as collecting statistics and plotting the results.
- Propose a possible RFC.

- Make a public release and a publication.
- Implement some fault management functions such as alarms reporting.
- Look into XML in Network Management (possibly for XML-RPC).
- Look more in detail at Java and network management, JMX (right now through AdventNet).
- Distributed Management of different DMARF nodes from various locations.
- Management of Grid-based Computing in DMARF.
- Analysis of CORBA and where it fits in Network Management in DMARF.
- Multimedia Management using SNMP.

3.3 Acknowledgments

- Dr. Chadi Assi
- SimpleWeb
- Open-Source Community
- AdventNet

Bibliography

- [Adv07a] AdventNet. *AdventNet SNMP Agent Toolkit Java Edition 6*. adventnet.com, 2007. <http://www.adventnet.com/products/javaagent/index.html>.
- [Adv07b] AdventNet. *AdventNet SNMP API 4*. adventnet.com, 2007. <http://snmp.adventnet.com/index.html>.
- [Ass07] Chadi Assi. *INSE7120: Advanced Network Management, Course Notes*. CIISE, Concordia University, 2007. <http://users.encs.concordia.ca/~assi/courses/inse7120.htm>.
- [HKS99] Harrie Hazewinkel, Carl W. Kalbfleisch, and Juergen Schoenwaelder. WWW Service MIB Module, RFC2594. IETF Application MIB Working Group, 1999. <http://www.simpleweb.org/ietf/mibs/modules/IETF/txt/WWW-MIB>.
- [(IA07] Internet Assigned Numbers Authority (IANA). *PRIVATE ENTERPRISE NUMBERS: SMI Network Management Private Enterprise Codes*. iana.org, March 2007. <http://www.iana.org/assignments/enterprise-numbers>.
- [MCSN03] Serguei Mokhov, Ian Clement, Stephen Sinclair, and Dimitrios Nicolacopoulos. Modular Audio Recognition Framework. Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, 2002–2003. Project report, <http://marf.sf.net>, last viewed April 2008.

- [Mok06] Serguei A. Mokhov. On design and implementation of distributed modular audio recognition framework: Requirements and specification design document. [online], August 2006. Project report, <http://arxiv.org/abs/0905.2459>, last viewed May 2009.
- [The07] The SimpleWeb. MIB module validation. [simpleweb.org](http://www.simpleweb.org), 2007. <http://www.simpleweb.org/ietf/mibs/validate/>.
- [The09] The MARF Research and Development Group. The Modular Audio Recognition Framework and its Applications. SourceForge.net, 2002–2009. <http://marf.sf.net>, last viewed December 2008.

Index

API

- adFeatures, 20, 23
- adSample, 15
- bRemoveNoise, 20
- bRemoveSilence, 20
- dSilenceThreshold, 20
- enterprises, 10, 14
- iFormat, 15
- lpcServiceEntry, 28
- oFeatureSet, 20
- oResultSet, 23
- preprocessingServiceEntry, 20
- preprocessingServiceTable, 20
- Sample, 20
- sampleLoadingServiceEntry, 15
- sampleLoadingServiceTable, 15
- SpeakerIdentApp, 2
- WwwService, 10

Applications, 24

Classification, 20

Files

- ATM-TC-MIB, 10
- MARF-APPS-LANGIDENTAPP.mib, 14
- MARF-APPS-SPEAKERIDENTAPP.mib, 14

- MARF-classification.mib, 14
- MARF-feature-extraction.mib, 14, 28
- MARF-MIB.mib, 14
- MARF-preprocessing.mib, 14
- MARF-sample-loading.mib, 14
- MARF-services.mib, 14
- MARF-storage.mib, 14
- MARF-types.mib, 14
- marf/doc/presentations/inse7120, 31
- marf/doc/src/graphics/distributed/mib, 31
- marf/doc/src/tex/inse7120, 31
- marf/src, 31
- marf/src/marf/net/snmp, 14, 31
- marf/src/mib, 10, 14
- WWW-MIB, 10

Introduction, 2

MARF

- Core Pipeline, 3
- Distributed Pipeline, 4
- Methodology, 9